applied sciences

MDPI

Article

Environment Classification for Unmanned Aerial Vehicle Using Convolutional Neural Networks

Carlos Villaseñor [1], Alberto A. Gallegos [2], Javier Gomez-Avila [1], Gehová López-González [2], Jorge D. Rios [1] and Nancy Arana-Daniel [1,*]

[1] Department of Computer Science, University of Guadalajara, 1421 Marcelino García Barragán, Guadalajara 44430, Jalisco, Mexico; carlos.villasenor@academicos.udg.mx (C.V.); jenrique.gomez@academicos.udg.mx (J.G.-A.); jorge.rarranaga@academicos.udg.mx (J.D.R.)

[2] Department of Artificial Intelligence, Hydra Technologies of Mexico, 6503 Vallarta Eje Poniente, Guadalajara 45010, Jalisco, Mexico; agallegos@hydra-technologies.com (A.A.G.); glopez@hydra-technologies.com (G.L.-G.)

* Correspondence: nancy.arana@academicos.udg.mx

Received: 18 June 2020; Accepted: 16 July 2020; Published: 20 July 2020

Featured Application: The approach presented in this paper is implemented in an autonomous UAV to provide the ability to change its path according to ground position and weather conditions, since sustaining an aircraft when flying through a dense cloud is not possible.

Abstract: Environment classification is one of the most critical tasks for Unmanned Aerial Vehicles (UAV). Since water accumulation may destabilize UAV, clouds must be detected and avoided. In a previous work presented by the authors, Superpixel Segmentation (SPS) descriptors with low computational cost are used to classify ground, sky, and clouds. In this paper, an enhanced approach to classify the environment in those three classes is presented. The proposed scheme consists of a Convolutional Neural Network (CNN) trained with a dataset generated by both, an human expert and a Support Vector Machine (SVM) to capture context and precise localization. The advantage of using this approach is that the CNN classifies each pixel, instead of a cluster like in SPS, which improves the resolution of the classification, also, is less tedious for the human expert to generate a few training samples instead of the normal amount that it is required. This proposal is implemented for images obtained from video and photographic cameras mounted on a UAV facing in the same direction of the vehicle flight. Experimental results and comparison with other approaches are shown to demonstrate the effectiveness of the algorithm.

Keywords: cloud detection; superpixel segmentation; convolutional neural networks; support vector machines

1. Introduction

Unmanned Aerial Vehicles (UAVs) have gained popularity in the last decades due to their capability for moving in three-dimensional space. UAVs were first used for military purposes. However, they are now used for surveillance, research, monitoring, and search and rescue activities [1]. These kinds of vehicles are suited for situations that are too dangerous and hazardous where direct monitoring is not humanly possible [2].

One of the challenges of UAV is the loss of communication with the remote pilot. For this reason, it is necessary to provide the vehicle with a certain level of autonomy to maintain flight in such scenarios. A UAV must be able to adapt and change its path according to ground position and weather conditions, since sustaining an aircraft when flying through a dense cloud is not possible [3].

Given weather indicators that allow the detection of clouds, can be seen from long distances [4]; it is possible to develop an intelligent system capable of avoiding them.

Cloud detection is a very challenging task; each big water cluster has a unique amorphous shape, which is continuously changing; making it impossible to extract characteristic features to be tracked with some descriptor such as with Speeded Up Robust Features (SURF) [5], then, other methods to extract information are needed, such as segmentation based on color, texture, and illumination [6–9].

In Reference [10], several simple-to-implement descriptors with linear computational costs are presented, showing a good training and generalization. Results from a video camera mounted on a UAV reported satisfactory results for two and three class classification in real-time.

Our proposed scheme describes and implements an approach to classify three elements of the environment (ground, sky, and clouds), using Superpixel Segmentation (SPS) and Support Vector Machine (SVM) to pre-train a Convolutional Neural Network (CNN), which is a form of deep learning model, trained end-to-end from raw pixel intensity values to classifier outputs. The spatial structure of the images makes it suitable to work with this kind of networks, setting connectivity between the filters (or layers) and the parameter sharing, and discrete convolutions [11].

The used images in this work were captured by a camera mounted on a UAV provided by Hydra Technologies of Mexico®; an example of an obtained image is shown in Figure 1.

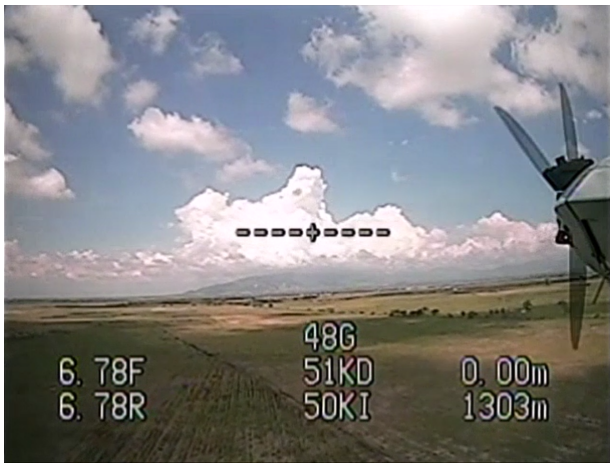


Figure 1. Captured image of a video stream from a camera mounted on an unmanned aerial vehicle (UAV). The image resolution is 720 width and 480 height at a 30 frames per second.

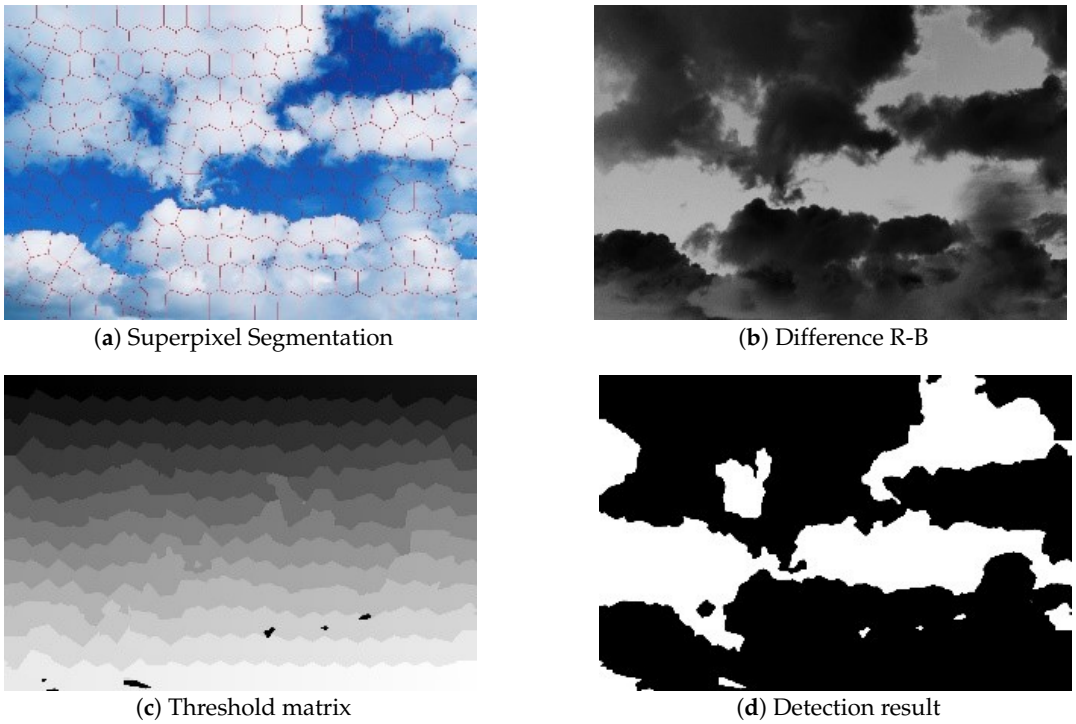
The outline of this papers is as follows: In Section 2, related work is presented. Section 3 presents a brief description of the used SVM whose output is used to pre-train the CNN. Section 4 presents the descriptors based on SPS methodology. In Section 5, the CNN architecture is described. Experimental results are presented in Section 6 and important conclusions are discussed in Section 7.

2. Related Work

Most of the research done on cloud detection is ground-based, where clouds are captured with instruments that obtain continuous all-sky images at pre-defined time intervals [12,13]. For a UAV, it is impossible to keep these conditions since the update intervals of information need to be shorter. Moreover, algorithms should not have a high computational cost, because onboard computers may not have the same processing power and memory capabilities as an off-board station. Also, a computer with high processing power in a UAV would require a higher demand for energy, which would require batteries with higher capabilities increasing the UAV weight, affecting the fuel consumption of the aircraft negatively.

Other works solve the problem of object identification using an undirected graph [14]. Computing the graph association matrix could be computationally expensive; in the worst-case scenario, it is a problem of $O(n^2)$ complexity [7,14]. These approaches are not suitable for real-time applications

working with high definition images [9]. In Reference [13], an automatic cloud detection for all-sky images using SPS is presented; the result and implementation of this algorithm shown in Figure 2. It can be seen that even if it is a good approximation, some information is lost in the final result. Considering these results and the computational complexity of the algorithm, it may not be suitable for these kinds of applications.



(**a**) Superpixel Segmentation (**b**) Difference R-B

(**c**) Threshold matrix (**d**) Detection result

Figure 2. Steps of the algorithm described in Reference [11].

On the other hand, algorithms based on image matting [6,8,15] try to reduce computational complexity. These algorithms extract foreground objects in images, but they are not easy to implement and take long processing time [9]. In these approaches, the algorithm distinguishes only between two classes (sky or ground), and it is difficult to add more classes.

Recently, deep learning techniques have been used to solve many computer vision tasks [14,16–20]. In particular, CNNs are good image classifiers [21–26]. Approaches like the ones presented in References [27,28] use CNNs that are trained to predict a class for each pixel. In contrast, this paper employs a segmentation on top of a CNN to label these clusters of pixels as the clean sky, clouds and ground.

3. Support Vector Machines

Vapnik introduced support vector machines in 1995, and they are widely used in classification tasks because of its simplicity and the convexity of the function to optimize [29]. Classification is treated as an optimization problem; the aim is to minimize a risk function R and maximize the separation between classes as represented by

$$w^* = \underset{w \in \Re^D}{\arg\min}\, F(w) = \frac{1}{2}\|w\|^2 + \zeta R(w), \tag{1}$$

where w is a normal vector orthogonal to the separating hyperplane, $\frac{1}{2}\|w\|^2$ is a quadratic regularization term, and $\zeta > 0$ is a fixed constant that limits the risk function. Equation (1) can be expressed using Lagrange multipliers as follows

$$\begin{aligned} \arg\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \psi_i \psi_j \Omega\left(\beta_i, \beta_j\right) \\ w^* = \sum_{i=1}^{n} \alpha_i \psi_i \beta_i, \end{aligned} \tag{2}$$

subject to $0 \leq \alpha_i \leq \zeta$ and $\sum_{i=1}^{n} \alpha_i \psi_i = 0$. Where $(\beta_i, \psi_i)_i^n$ is a training set, from which β_i is an n-dimensional input vector and ψ_i its corresponding label. Notice that α_i are Lagrange multipliers and $\Omega(\beta_i, \beta_j)$ is the value of the kernel matrix Ω defined by the inner product $\phi\left(\beta_i\right) \cdot \phi\left(\beta_j\right)$, where ϕ is a non-linear mapping to a high dimensional space. The advantage of using this dual formulation is the use of kernels that introduce the feature space by implicitly mapping the input data into a higher-dimensional space where non-linearly separable data can be linearly separable [30,31].

A CNN requires a massive amount of training data; this task is usually tedious for a human. In that sense, the data used to pre-train the network has been created by an human expert and a SVM that classifies an image segmented with superpixels, that is, sub-areas represented by only a descriptor instead of having several values for every pixel in the sub-area.

4. Descriptors

Most of the descriptors are developed to classify only two classes and cannot be naturally scaled to m different classes. The descriptors presented in this section have linear complexity $O(n)$, and a descriptor capable of increasing the number of classes to three is proposed.

4.1. Descriptors Based on Superpixel Segmentation and Histogram

In this section, three descriptors that use their histograms as features are described. Three images must be obtained to construct the required descriptors. Let (R, G, B) be the channels red, green, and blue, respectively; the descriptors will be obtained from $R - B$, R/B, and RGB images. Cloud detection algorithms commonly use color to determine if a region of the image is a cloud. Cloud particles have a similar dispersion of B and R intensity, whereas clear sky presents more B than R intensity [12,13].

For N pixels, M superpixels will be generated based on color similarity and proximity using Simple Linear Iterative Clustering (SILC) [32] in CIELAB color space. SILC initializes M clusters centers $C_m = [l_m, a_m, b_m, x_m, y_m]^T$ on a regular grid space, where (l, a, b) is the color vector in CIELAB space and (x, y) are the pixel coordinates. Each superpixel has an approximate size of N/M and the center will be located every $S = \sqrt{N/M}$.

SILC computes a distance D between pixel i and its nearest cluster center C_m

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{s}\right) r^2}, \tag{3}$$

where $r \in [1, 40]$ is a constant that allows pondering between color similarity and spatial proximity, d_c and d_s are defined by

$$d_c = \sqrt{\left(l_j - l_i\right)^2 + \left(a_j - a_i\right)^2 + \left(b_j - b_i\right)^2} \tag{4}$$

$$d_s = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2}. \tag{5}$$

The clusters are adjusted to take the value of the main vector of the pixels in C_m, and a residual error E between the new cluster center and previous centers is computed using L_2 norm. The algorithm stops when E reaches a certain threshold.

The descriptor β of the superpixel k is obtained from a histogram of 16 values for each superpixel in $R-B$ and R/B images. The intensity value of pixel $i \in k$ is divided by 16 and rounded downward to its nearest integer value. In the case of the RGB image, a histogram for each channel is obtained.

4.2. Superpixel Segmentation with Gabor Filter

For this approach, a pre-processing step is needed and is showed in Figure 3. Since clouds enhance the $R-B$ difference, this image has been used, and its histogram has been normalized. Gaussian blur has been applied to reduce noise, before the binarization with Otsu's method [33], which is obtained by solving

$$\sigma_b^2(t) = P_0 P_1 (\mu_0 - \mu_1)^2, \tag{6}$$

where P_0 and P_1 are class probabilities obtained from a histogram L and separated by a threshold t; and μ_0 and μ_1 are the means of the classes. This is represented by Equations (7)–(10):

$$P_0(t) = \sum_{i=0}^{t-1} p(i) \tag{7}$$

$$P_1(t) = \sum_{i=t}^{L-1} p(i) \tag{8}$$

$$\mu_0(t) = \sum_{i=0}^{t-1} i \frac{p(i)}{P_0} \tag{9}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} i \frac{p(i)}{P_1}. \tag{10}$$

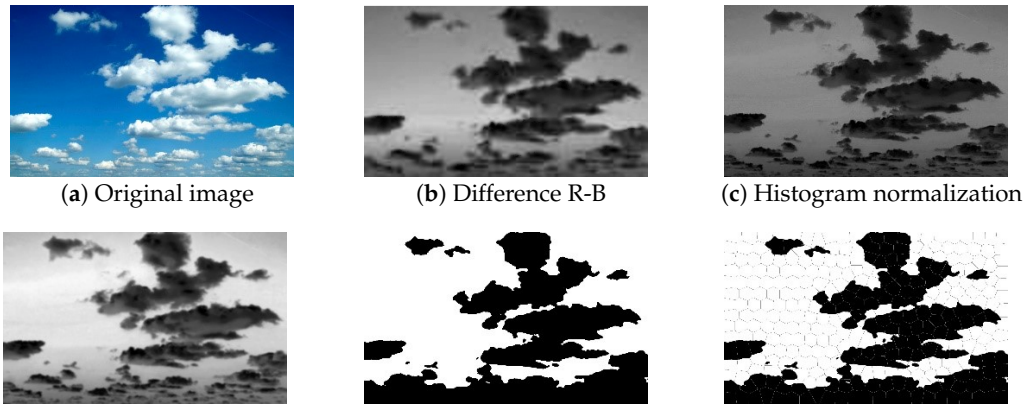
At this step, it is easy to classify clean-sky from clouds; however, as can be seen in Figure 4, it is not possible to make a distinction between clouds and ground. Because of this, it is necessary to use another descriptor capable of distinguishing between them. In this case, the Gabor filter [34] is applied to the original image to get the descriptor because of its ability to permit texture representation and discrimination. The filter has a strong response with structures in the image that have the same direction [35]. The following two-dimensional Gabor functions are used:

$$g_{\lambda,\Theta,\rho}(x,y) = e^{-\left(\left(x'^2+\gamma^2 y'^2\right)/2\sigma^2\right)} \cos\left(2\pi\frac{x'}{\lambda}+\rho\right). \tag{11}$$

$$x' = x\cos\Theta + y\sin\Theta \tag{12}$$

$$y' = -x\cos\Theta + y\sin\Theta, \tag{13}$$

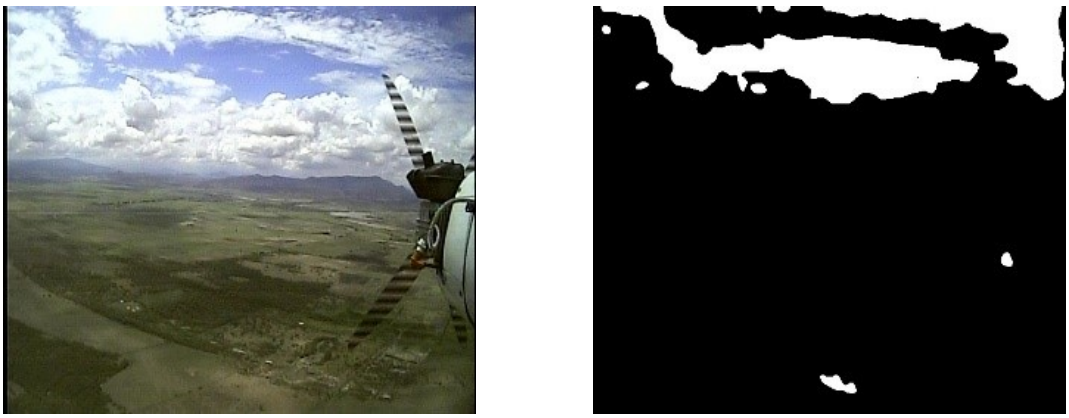
where λ is the wavelength, Θ is the orientation, ρ is the phase offset, γ is the aspect ratio, and $\sigma = 0.56\lambda$ is the standard deviation.



(**a**) Original image (**b**) Difference R-B (**c**) Histogram normalization

(**d**) Gaussian blur (**e**) Otsu's binarization (**f**) Superpixel Segmentation

Figure 3. Steps to generate the proposed descriptor for three classes (ground, cloud and sky).



(**a**) Original image (**b**) Binary image

Figure 4. Clouds and ground classes cannot be easily distinguished.

Four Gabor filters are calculated for $\Theta \in (\pi/4, \pi/(2, 3\pi/(4, \pi)))$. The filtered images are converted to grayscale, and the mean of the values of the image is added to the descriptor. The variance of superpixel k, in each Gabor filtered image is calculated and added to the descriptor β_k. Moreover, spatial information has been included in the descriptor since ground superpixels will have lower spatial values, while clouds superpixels will have higher spatial values.

5. Convolutional Neural Networks

CNNs are commonly used for processing data contained in a matrix or grid, such as images, that are represented by a 2D matrix. Their name comes from the mathematical operation called convolution, which is an operation on two functions to produce a third function that expresses how one of them is modified by the other. In computer vision and image processing, the convolution operation is used to reduce noise and enhance features in images.

Let us suppose that $s(t)$ is the output of the convolution; the operation is given by

$$s(t) = \int l(a)\, h(t-a)\, da, \tag{14}$$

where function l is the output of a sensor (and usually referred to as the input in CNN terminology), h is a weighting function (also known as the kernel), a is the age of the measurement. The convolution is commonly denoted with an asterisk as follows

$$s(t) = (l * h)(t). \tag{15}$$

This data is usually discretized, and if time t can only take integer values then it is possible to define the convolution as a discrete operation as follows

$$s(t) = \sum_{a=-\infty}^{\infty} l(a)\, h(t-a). \tag{16}$$

The input and the output are multidimensional arrays, and every element must be explicitly stored separately. It is assumed that every element out of the set of points, for which the values are stored, is zero; therefore, the infinite summation can be implemented over a finite number of array elements, and also, it can be used over more than one axis at a time. Let I be a two-dimensional image, K a two-dimensional kernel, the convolution for images is given by

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)\, K(i-m, j-n) \tag{17}$$

and can graphically be described, as shown in Figure 5.

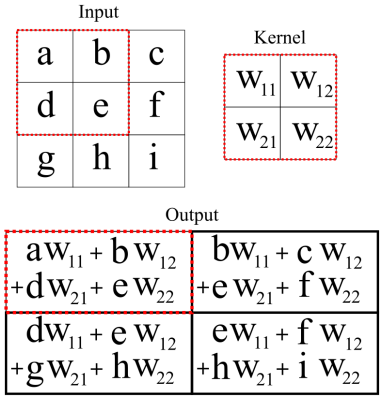


Figure 5. Graphical description of convolution operation.

The convolution presents two properties that can help to improve a machine learning system—sparse interactions and parameter sharing [36].

Due to its sparse interactions, it is necessary to store fewer parameters and fewer operations; however, units in the deeper layers may indirectly interact with a more significant portion of the input and describe more complicated interactions between pixels, as described in Figure 6.

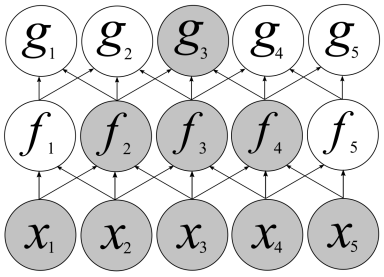


Figure 6. Description of sparse interactions. Even if direct connections seem to be sparse, more units at deeper layers are indirectly connected.

A CNN consist of three steps. First, several convolutions in parallel produce a set of linear activations. Then, a detector step is implemented, where nonlinear activation functions take the linear activations as the argument. Finally, a pooling function is used to modify the output of the layer, making the representation invariant to small translations of the input [36].

Environment Classification with CNN

CNNs have demonstrated effectiveness in image recognition, segmentation, and detection [11]. The architecture of the network is shown in Figure 7. Each layer uses a Rectified Linear Unit (ReLU) function for their activation; except for the last one, whose activation function is a sigmoid, and is given by $f(x) = 1/(1 + e^{-x})$.

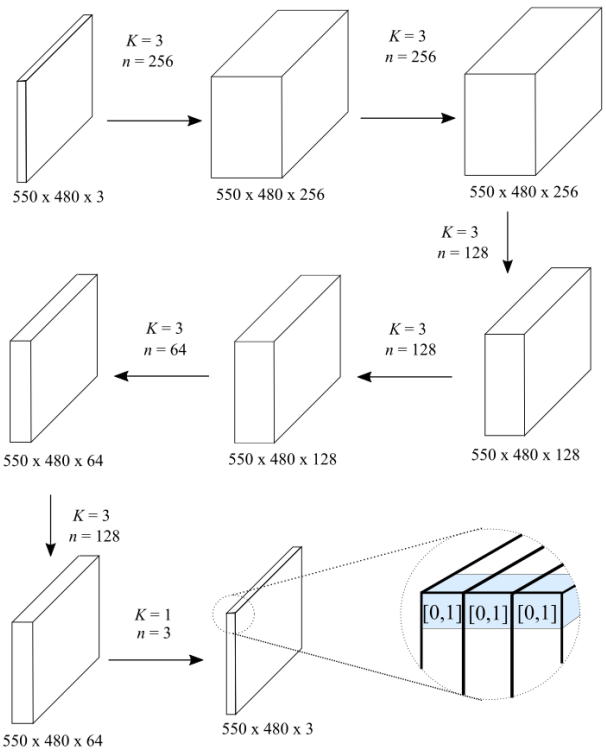


Figure 7. Convolutional Neural Network (CNN) architecture. In all cases stride = 1 and a zero-padding = 1. The output image has three channels (for sky, cloud, and ground), and each pixel is labeled based on its three channels values.

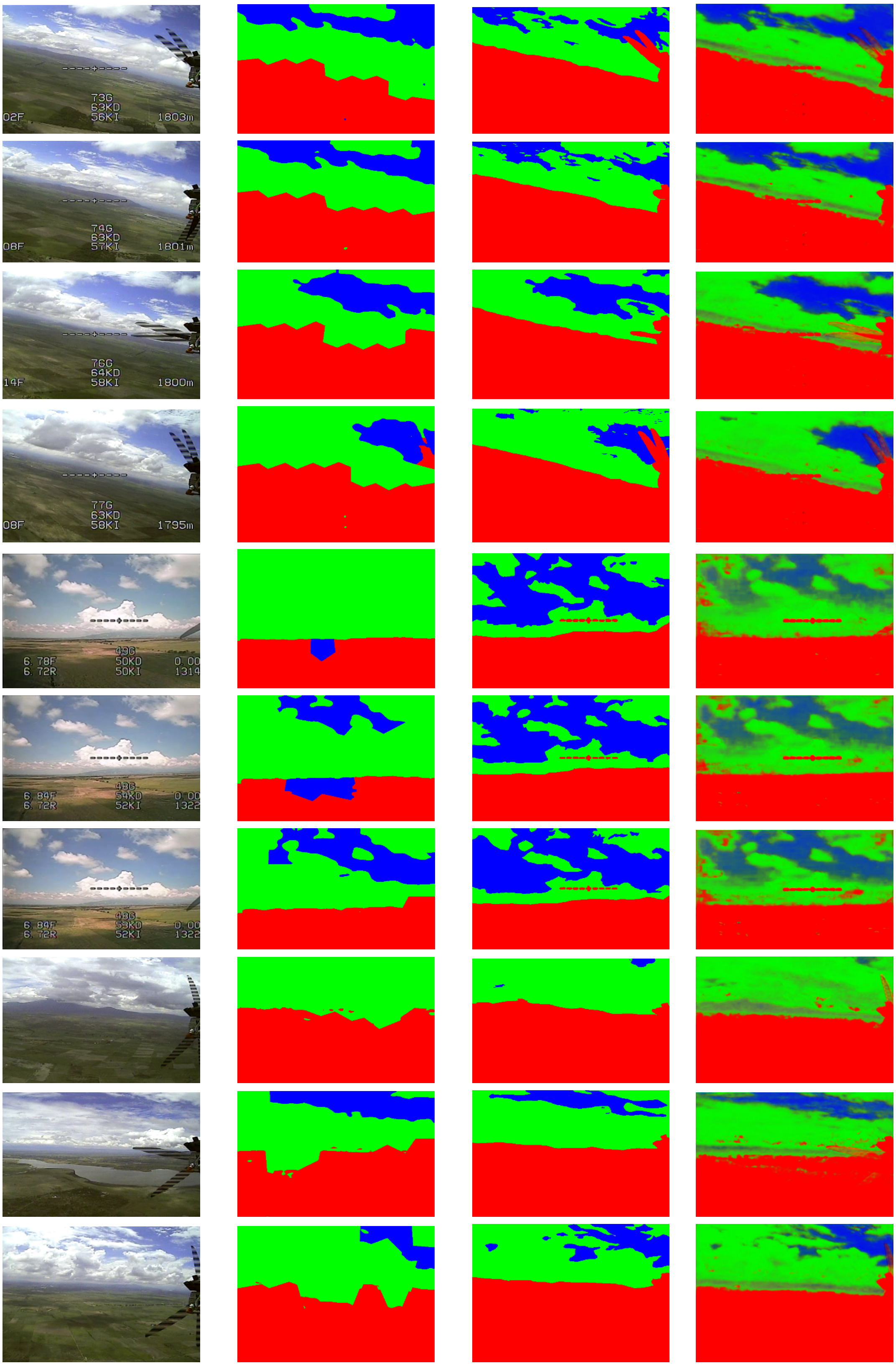
CNN is a class of deep learning model that requires a large quantity of data to be trained. In practice, it is relatively rare to access large data sets, and it is a tedious task for a human to generate them [21]. In this work, one part of this data is generated by the classification of the superpixels made by the SVM; nevertheless, training the CNN only with SVM information would make the CNN learn from a Support Vector Machine. Another set of training data was provided by a human expert to avoid this behavior. Finally, the training data were artificially enlarged using data augmentation.

6. Experimental Results

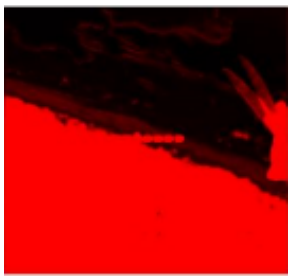
In this section, results of proposal are presented, the pre-train step is carried out with 1000 images provided by an SVM. Then, only twenty ground truth images classified by a human expert are used for supervised training. Table 1 shows ten test images used to demonstrate the effectiveness of the proposed algorithm. These photos were taken from three different flights at a fixed altitude, but different in each flight, and different weather conditions. Although they are not consecutive frames, pictures from rows 5 to 7 were taken from a straight and level flight; and there is little difference between them, however, the SPS-SVM clearly presents a different classification between these images. Additionally, the data set is artificially enlarged, applying geometric transformations to the training set.

Table 1. Test results. The first column shows the original image. The second column shows the Superpixel Segmentation (SPS)–Support Vector Machine (SVM) classification. The ground truth, generated by a human is shown in the third column. In the fourth column, the classification made by the CNN is presented.

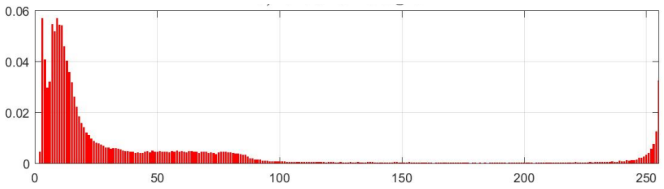
Original Image	SPS-SVM Classification	Human Labeled	CNN Output



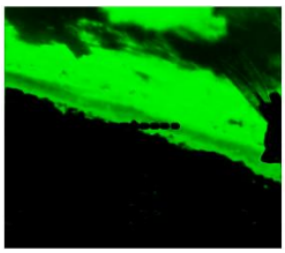
For each pixel, the CNN outputs the probability of belonging to each class. By using these probabilities as pixel intensities, we form grayscale images in Figure 8. Their histogram are also shown. Moreover, the probabilities of each class are scaled and presented in Figure 9 to demonstrate which pixels activate the output layer for each class.



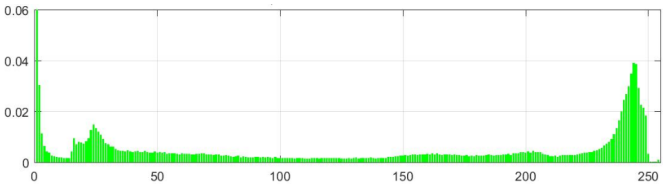
(**a**) Ground



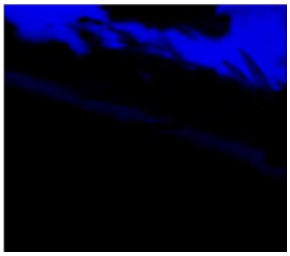
(**b**) Ground histogram



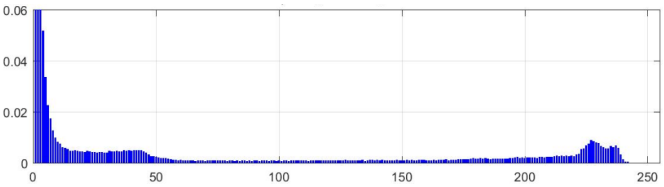
(**c**) Clouds



(**d**) Clouds histogram



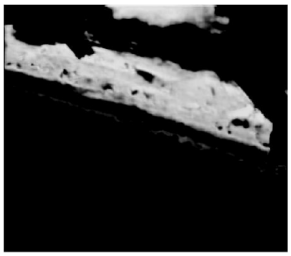
(**e**) Sky



(**f**) Sky histogram

Figure 8. Pixel distribution for each class.

(**a**) Scaled ground



(**b**) Scaled clouds

(**c**) Scaled sky

Figure 9. Scaled probabilities for each class.

To display a better visualization of the performance, Table 2 shows the confusion matrices of both approaches. These matrices compare the prediction of the algorithm with the ground truth. The closer it gets to an identity matrix, the less the algorithm gets confused between classes.

Table 2. Confusion Matrices comparison. (S: Sky, C: Cloud, G: Ground).

Test	SPS-SVM vs. Human	CNN vs. Human
1	Predicted / True: S: 0.983 0.017 0.000; C: 0.202 0.798 0.000; G: 0.015 0.223 0.763 / S C G	Predicted / True: S: 0.976 0.024 0.000; C: 0.149 0.848 0.003; G: 0.013 0.116 0.871 / S C G
2	Predicted / True: S: 0.979 0.021 0.000; C: 0.176 0.824 0.000; G: 0.000 0.159 0.841 / S C G	Predicted / True: S: 0.972 0.002 0.000; C: 0.134 0.863 0.003; G: 0.000 0.098 0.902 / S C G
3	Predicted / True: S: 0.903 0.097 0.000; C: 0.058 0.917 0.025; G: 0.000 0.130 0.872 / S C G	Predicted / True: S: 0.897 0.103 0.000; C: 0.039 0.912 0.048; G: 0.000 0.085 0.915 / S C G
4	Predicted / True: S: 0.878 0.121 0.001; C: 0.030 0.962 0.008; G: 0.019 0.121 0.860 / S C G	Predicted / True: S: 0.854 0.146 0.000; C: 0.013 0.983 0.004; G: 0.016 0.090 0.894 / S C G
5	Predicted / True: S: 0.000 1.000 0.000; C: 0.000 1.000 0.000; G: 0.039 0.129 0.832 / S C G	Predicted / True: S: 0.192 0.806 0.002; C: 0.000 0.907 0.093; G: 0.000 0.038 0.962 / S C G
6	Predicted / True: S: 0.282 0.718 0.000; C: 0.002 0.998 0.000; G: 0.127 0.148 0.724 / S C G	Predicted / True: S: 0.271 0.728 0.001; C: 0.000 0.915 0.085; G: 0.000 0.055 0.945 / S C G
7	Predicted / True: S: 0.524 0.476 0.000; C: 0.011 0.983 0.006; G: 0.000 0.129 0.871 / S C G	Predicted / True: S: 0.418 0.569 0.013; C: 0.000 0.892 0.108; G: 0.000 0.048 0.952 / S C G
8	Predicted / True: S: 0.000 0.999 0.001; C: 0.000 0.995 0.005; G: 0.000 0.103 0.897 / S C G	Predicted / True: S: 0.087 0.912 0.001; C: 0.000 0.984 0.016; G: 0.000 0.127 0.873 / S C G
9	Predicted / True: S: 0.975 0.025 0.000; C: 0.153 0.829 0.018; G: 0.000 0.122 0.878 / S C G	Predicted / True: S: 0.922 0.077 0.001; C: 0.035 0.940 0.026; G: 0.000 0.100 0.900 / S C G
10	Predicted / True: S: 0.547 0.453 0.000; C: 0.024 0.976 0.000; G: 0.000 0.127 0.873 / S C G	Predicted / True: S: 0.658 0.313 0.028; C: 0.006 0.989 0.006; G: 0.000 0.098 0.902 / S C G

As seen in Table 1, adding a few images from an human expert, avoids CNN to behave as an SVM. The advantage is that an human expert need to generate only twenty training images which

the network can make a good generalization and correct mistakes generated by the SVM, for example, rows 5 and 6 in Table 1.

From the matrices in Table 2, recall, precision, and F1 score are computed to measure the effectiveness of the algorithm and to compare it with the SPS-SVM. These results are shown in Table 3. There is no entry for SVM in test 8 because, in such an experiment, only two classes were found (missing sky).

The confusion matrices for both techniques are very close. To get a better understanding for each matrix the macro versions of the recall, precision, and f1-score, in Table 3. In Figure 10, the same score to get a better visual understanding of proposal performance is plotted. The proposal overcome the SPS-SVM in almost all the samples (except for the sample seven).

Table 3. The measure of the test for both approaches. Bold letters denote the winner technique

Test	Approach	Recall	Precision	F1 Score
1	SVM	0.8497	0.7962	0.8059
	CNN	**0.8975**	**0.8750**	**0.8801**
2	SVM	0.8789	0.8503	0.8563
	CNN	**0.9113**	**0.8973**	**0.9006**
3	SVM	0.9024	0.8886	0.8914
	CNN	**0.9172**	**0.9121**	**0.9134**
4	SVM	0.9128	0.8964	0.8989
	CNN	**0.9321**	**0.9225**	**0.9238**
5	SVM	0.5127	0.5985	0.5193
	CNN	**0.8345**	**0.6916**	**0.6522**
6	SVM	0.7417	0.6413	0.6314
	CNN	**0.8419**	**0.7063**	**0.6833**
7	SVM	**0.8767**	**0.7751**	**0.7818**
	CNN	0.8470	0.7479	0.7413
8	SVM	-	-	-
	CNN	**0.9231**	**0.9099**	**0.9087**
9	SVM	0.8919	0.8682	0.8741
	CNN	**0.9224**	**0.9161**	**0.9173**
10	SVM	0.9016	0.8786	0.8792
	CNN	**0.9247**	**0.9161**	**0.9173**

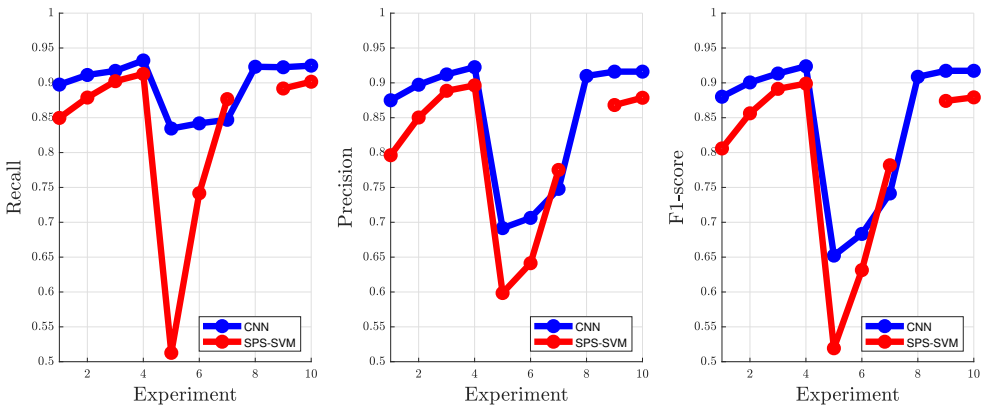


Figure 10. The measure of the test for both approaches.

For both schemes, we find the hyperparameters heuristically guided by the train and test scores obtained from 30 executions episodes. Finally, in this paper, we do not show a run-time comparison because CNN was implemented in TensorFlow-Keras Framework; consequently, it runs over the Graphical Process Unit (GPU). On the other hand, the system SPS-SVM was implemented as a sequential algorithm to be executed on the CPU due to the complexity of its parallelization. CNN has lower run-time than SPS-SVM, but the comparison is not fair until we get a parallelized implementation of SPS-SVM.

7. Conclusions

As can be seen in the previous section, the approach gives good results not only classifying the parts of the environment that are desired to be segmented into classes but also reducing the tedious labor of generating a data set by human hand. As seen on the results image, the proposal can classify with more detail than a SVM or a human using basic image editing tools.

The CNN for pixel classification commonly needs a big data set to train; in this paper, a CNN is pre-trained with the prediction of an SPS–SVM. Then, the SPS–SVM can be considered as a data augmentation process to generate synthetic labeled data.

The approach is fast enough to provide sensitive information in a short time, so a UAV can take decisions with recent information. Future work will focus on improving the classification by adding estimations on the different types of clouds that can be found in the environment and the risk they could represent for a UAV.

Author Contributions: Conceptualization, C.V. and N.A.-D.; methodology, C.V.; software, A.A.G.; validation, G.L-G., C.V. and J.G.-A.; formal analysis, N.A.-D. and A.A.G.; investigation, J.D.R.; writing—original draft preparation, J.G.-A.; writing—review and editing, G.L.-G., J.D.R. and J.G.-A.; visualization, A.A.G. and G.L.-G.; supervision, J.D.R.; project administration, C.V. and N.A.-D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONACYT México grants numbers CB256769, CB258068, and PN-4107.

Acknowledgments: The authors would like to thank Hydra Technologies de México for providing the data for the development of this work.

Conflicts of Interest: The authors declare no conflict of interest. The founders 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.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks
UAV	Unmanned Aerial Vehicle
ReLU	Rectified Linear Unit
SILC	Simple Linear Iterative Clustering
SPS	Superpixel Segmentation
SURF	Speeded Up Robust Features
SVM	Support Vector Machine

References

1. Millbrooke, A. *Aviation History*; Jeppesen: Englewood, CO, USA, 2006.
2. Gupta, S.G.; Ghonge, D.; Jawandhiya, P.M. Review of unmanned aircraft system (UAS). *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)* **2013**, *2*. [CrossRef]
3. Bottyán, Z.; Tuba, Z.; Gyöngyösi, A.Z. Weather Forecasting System for the Unmanned Aircraft Systems (UAS) Missions with the Special Regard to Visibility Prediction, in Hungary. In *Critical Infrastructure Protection Research*; Springer International Publishing: Cham, Switzerland, 2016; pp. 23–34.
4. George, J.J. *Weather Forecasting for Aeronautics*; Academic Press: London, UK, 2014.

5. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **2008**, *110*, 346–359. [CrossRef]
6. Wang, J.; Cohen, M.F. Optimized color sampling for robust matting. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 1–8.
7. Ren, X.; Malik, J. Learning a classification model for segmentation. In Proceedings of the Ninth IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; IEEE: Piscataway, NJ, USA, 2003; p. 10.
8. Levin, A.; Lischinski, D.; Weiss, Y. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 228–242. [CrossRef] [PubMed]
9. Zhang, Q.; Xiao, C. Cloud detection of RGB color aerial photographs by progressive refinement scheme. *IEEE Trans. Geosci. Remote. Sens.* **2014**, *52*, 7264–7275. [CrossRef]
10. Peña-Olivares, O.; Villaseñor, C.; Gallegos, A.A.; Gomez-Avila, J.; Arana-Daniel, N. Automatic Environment Classification for Unmanned Aerial Vehicle Using Superpixel Segmentation. In Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Gudalajara, Mexico, 7–9 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
11. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
12. Ghonima, M.; Urquhart, B.; Chow, C.; Shields, J.; Cazorla, A.; Kleissl, J. A method for cloud detection and opacity classification based on ground based sky imagery. *Atmos. Meas. Tech. Discuss.* **2012**, *5*, 4535–4569. [CrossRef]
13. Liu, S.; Zhang, L.; Zhang, Z.; Wang, C.; Xiao, B. Automatic cloud detection for all-sky images using superpixel segmentation. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 354–358.
14. Boykov, Y.Y.; Jolly, M.P. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV 2001), Vancouver, BC, Canada, 7–14 July 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 1, pp. 105–112.
15. Xiao, C.; Liu, M.; Xiao, D.; Dong, Z.; Ma, K.L. Fast closed-form matting using a hierarchical data structure. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 49–62. [CrossRef]
16. Black, K.M.; Law, H.; Aldouhki, A.; Deng, J.; Ghani, K.R. Deep learning computer vision algorithm for detecting kidney stone composition. *BJU Int.* **2020**, *125*, 920–924. [CrossRef]
17. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
18. Wang, Z.; Chen, J.; Hoi, S.C. Deep learning for image super-resolution: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [CrossRef]
19. Arabi, S.; Haghighat, A.; Sharma, A. A deep-learning-based computer vision solution for construction vehicle detection. *Comput. Aided Civ. Infrastruct. Eng.* **2020**, *35*, 753–767. [CrossRef]
20. Wu, X.; Sahoo, D.; Hoi, S.C. Recent advances in deep learning for object detection. *Neurocomputing* **2020**, *395*, 39–64.
21. Attari, N.; Ofli, F.; Awad, M.; Lucas, J.; Chawla, S. Nazr-cnn: Fine-grained classification of uav imagery for damage assessment. In Proceedings of the 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, Japan, 19–21 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 50–59.
22. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; pp. 276–279.
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
24. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv* **2013**, arXiv:1312.6229.
25. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
26. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.

27. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
28. Eigen, D.; Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2650–2658.
29. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
30. Muller, K.R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **2001**, *12*, 181–201. [CrossRef]
31. Haykin, S. *Neural Networks: A Comprehensive Foundation*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1994.
32. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
33. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
34. Kruizinga, P.; Petkov, N. Nonlinear operator for oriented texture. *IEEE Trans. Image Process.* **1999**, *8*, 1395–1407. [CrossRef]
35. Grigorescu, S.E.; Petkov, N.; Kruizinga, P. Comparison of texture features based on Gabor filters. *IEEE Trans. Image Process.* **2002**, *11*, 1160–1167. [CrossRef] [PubMed]
36. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.